

FIG. 1

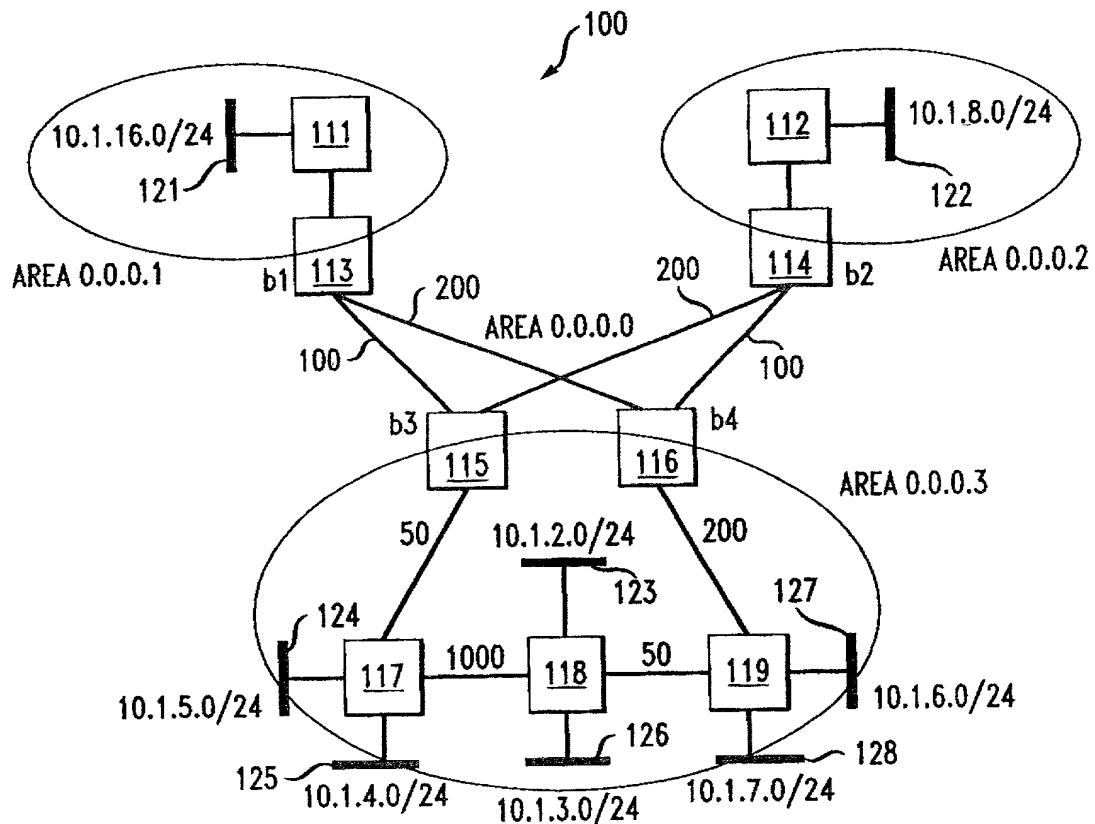


FIG. 2

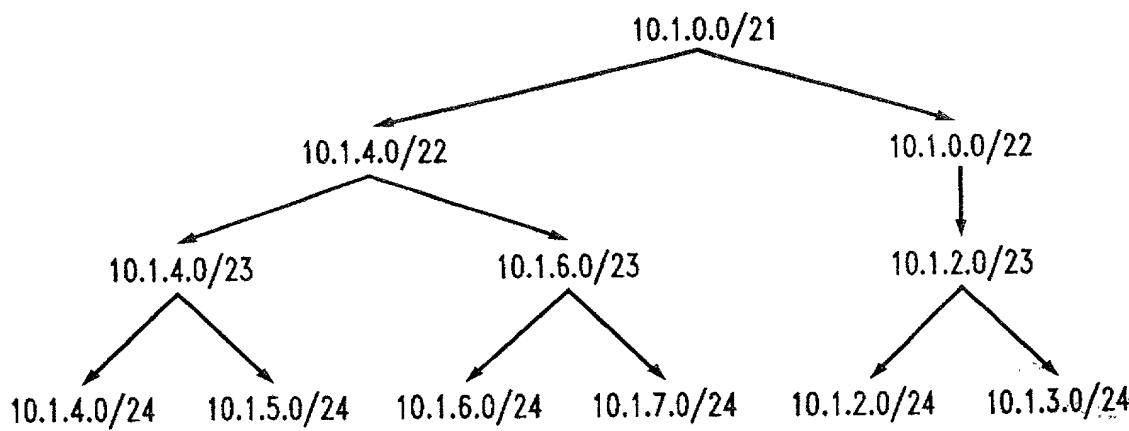


FIG. 3

```

procedure COMPUTEMINERROR(Aggregate x, Aggregate y, integer l)
1. if subTree[x, y, l].computed = true
2.   return [subTree[x, y, l].error, subTree[x, y, l].aggregates]
3. minError := minError1 := minError2 := ∞
4. if x is a leaf {
5.   minError1 :=  $\sum_{s \in S} D(s, t) * (lsp(s, x, \{y\}, W_A) - lsp(s, x))$ 
6.   if l > 0
7.     minError2 :=  $\sum_{s \in S} D(s, t) * (lsp(s, x, \{x\}, W_A) - lsp(s, x))$ 
8.   if minError1 ≤ minError2
9.     [subTree[x, y, l].error, subTree[x, y, l].aggregates] := [minError1, ∅]
10.  else
11.    [subTree[x, y, l].error, subTree[x, y, l].aggregates] := [minError2, {x}]
12.}
13. if x has a single child u {
14.   [minError1, aggregates1] := COMPUTEMINERROR(u, y, l)
15.   if l > 0
16.     [minError2, aggregates2] := COMPUTEMINERROR(u, x, l - 1)
17.   if minError1 ≤ minError2
18.     [subTree[x, y, l].error, subTree[x, y, l].aggregates] := [minError1, aggregates1]
19.   else
20.     [subTree[x, y, l].error, subTree[x, y, l].aggregates] := [minError2, aggregates2 ∪ {x}]
21.}
22. if x has children u and v {
23.   for i := 0 to l {
24.     [minError1, aggregates1] := COMPUTEMINERROR(u, y, i)
25.     [minError2, aggregates2] := COMPUTEMINERROR(v, y, k - i)
26.     if minError1 + minError2 < minError
27.       minError := minError1 + minError2
28.       aggregates := aggregates1 ∪ aggregates2
29.   }
30.   for i := 0 to l - 1 {
31.     [minError1, aggregates1] := COMPUTEMINERROR(u, x, i)
32.     [minError2, aggregates2] := COMPUTEMINERROR(v, x, k - i - 1)
33.     if minError1 + minError2 < minError
34.       minError := minError1 + minError2
35.       aggregates := aggregates1 ∪ aggregates2 ∪ {x}
36.   }
37.   [subTree[x, y, l].error, subTree[x, y, l].aggregates] := [minError, aggregates]
38.}
39. subTree[x, y, l].computed := true
40. return [subTree[x, y, l].error, subTree[x, y, l].aggregates]

```

FIG. 4

```

procedure COMBINEMINERROR()
1. for i = 1 to m
2.   for j = 0 to k {
3.      $T_i[j].[\text{error, aggregates}] := \text{COMPUTEMINERROR}(r(T_i), \in, j)$ 
4.      $X_i[j].[\text{error, aggregates}] := [\infty, \emptyset]$ 
5.   }
6.   for j = 0 to k
7.      $X_i[j].[\text{error, aggregates}] := T_i[j].[\text{error, aggregates}]$ 
8.   for i = 1 to m
9.     for j = 0 to k
10.      for l = 0 to j
11.        if ( $X_{i-1}[l].\text{error} + T_i[j-l].\text{error} < X_i[j].\text{error}$ ) {
12.           $X_i[j].\text{error} = X_{i-1}[l].\text{error} + T_i[j-l].\text{error}$ 
13.           $X_i[j].\text{aggregates} = X_{i-1}[l].\text{aggregates} \cup T_i[j-l].\text{aggregates}$ 
14.        }

```

FIG. 5

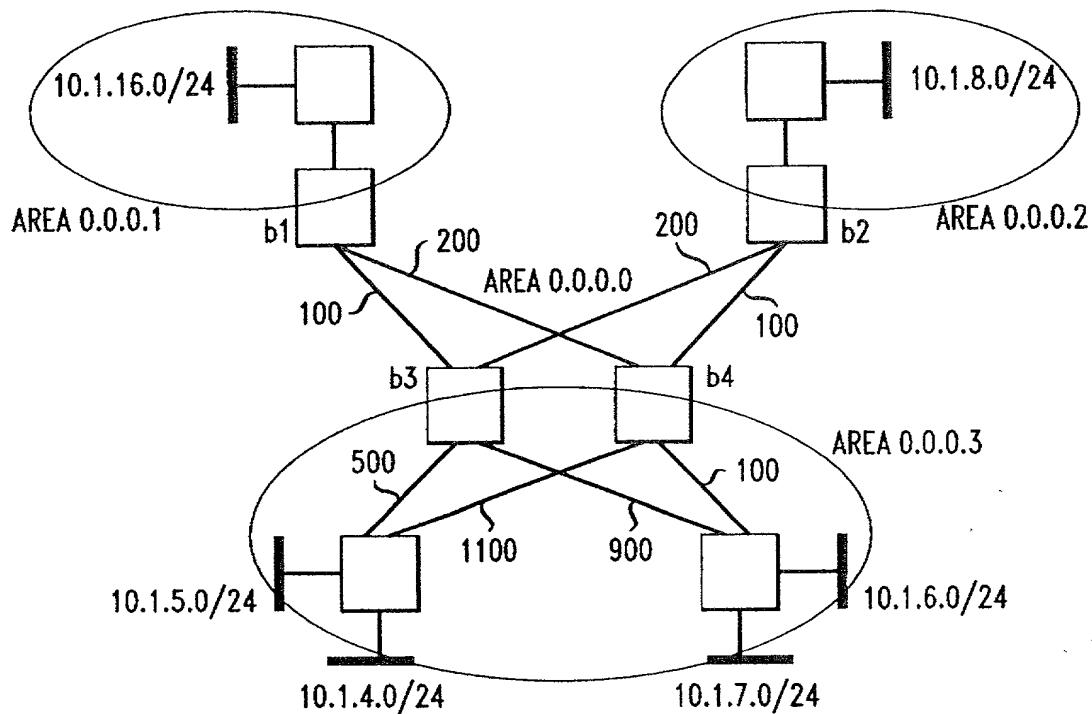


FIG. 6

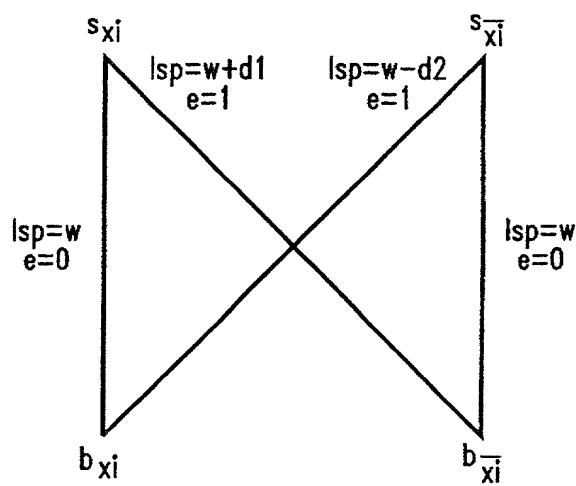


FIG. 7A

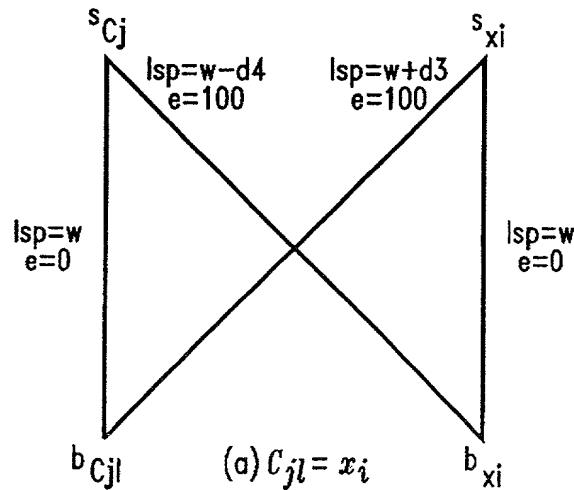


FIG. 7B

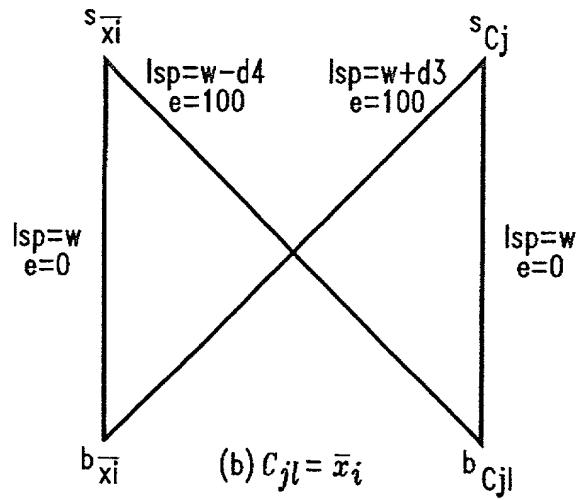


FIG. 8

```

procedure COMPUTEWIGHTSCUMULATIVE ()
1. for each  $b \in B_i$  set  $W_{min}(b) := 0$ 
2. for  $i := 1$  to  $r$  {
3.    $W := W_{min}$ 
4.   Choose a random subset  $R \subseteq B_i$  of ABRs
5.   for each  $b \in R$  set  $W(b)$  to a random weight in  $[0, L]$ 
6.   if  $\sum_{s \in se} (s, B(s, W)) < \sum_{s \in se} (s, B(s, W_{min}))$ 
7.      $W_{min} := W$ 
8.   }
9. return  $W_{min}$ 

```

FIG. 9

```

procedure ComputeWeightsMax(Q)
1. for each  $b \in B_i$  set  $W_{old}(b) := 0$ 
2. while ( $Pb_2B$ 
  i  $W_{old}(b) \leq ($ 
  j  $B_i j * (j B_i j - 1)$ 
  2 ) * lspmax) f3. Let

```

Q0 be a new set of inequalities that result when the value $W_{old}(b)$ is substituted for each variable $W(b)$ only on the LHS of each inequality in Q. 4. Set $W_{new}(b)$ to the smallest possible value such that each inequality in Q0 is satisfied when $W_{new}(b)$ is substituted for variable $W(b)$ in Q0 5. **if** $W_{new} = W_{old}$ 6. **return** W_{new} 7. **else** 8. $W_{old} := W_{new}$ 9. **g** 10. **return** "there does not exist a weight assignment W"

FIG. 10

```

procedure COMPUTEWIGHTSTWOABR()
1. Set  $V_{opt} := v(s_1)$ ,  $E := E_{opt} := \sum_{s \in se} (s, b_1)$ 
2. for  $j := 1$  to  $n$  {
3.    $E := E + e(s_j, b_2) - e(s_j, b_1)$ 
4.   if  $E < E_{opt}$ 
5.      $V_{opt} := v(s_{j+1})$ ,  $E_{opt} := E$ 
6.   }
7. return  $V_{opt}$ 

```